Real roots counting for some robotics problems

Fabrice Rouillier

REAL ROOT COUNTING FOR SOME ROBOTICS PROBLEMS

FABRICE ROUILLIER
IRMAR Université de Rennes I
Avenue des Buttes de Coesme, 35042 Rennes cedex, France

Abstract. We propose two algorithms to compute the number of real roots of zero-dimensional systems, using effective algebraic methods. To compare their behaviour on practical examples, we apply these methods to systems that describe some robotics problems (e.g. direct kinematic problem of parallel manipulators).

1. Introduction

Let Z be a domain, K its Fraction field, $S = \{f_1, f_2, \cdots, f_s\}$ a system of polynomial equations in $Z[X_1, \cdots, X_n]$ with a finite set of distinct roots $\mathcal{X} = \{\alpha_1, \ldots, \alpha_d\}$ of respective multiplicities $\{m_1, \ldots, m_d\}$ and , I the ideal generated by $\mathcal{S}$.

In such cases, $A = \frac{K[X_1,\cdots,X_n]}{I}$ is a finite dimensional vector space of dimension $D = \sum_{i=1}^{d} m_i$.

Referring to (Rouillier , 1995), we assume in this paper the existence of efficient algorithms that compute the following (from a Groebner basis) :

- A Z-basis $\mathcal{B} = \omega_1, \ldots, \omega_D$ of A
- The multiplication table of A (with respect to $\mathcal{B}$), defined by a $D \times D$ matrix $MT = (MT[i,j])_{1 \leq i,j \leq D}$, where $MT[i,j]$ is the column-vector whose coordinates are the coefficients of $\omega_i\omega_j$ with respect to $\mathcal{B}$ (e.g. $\omega_i\omega_j = \sum_{k=1}^{D}(MT[i,j])_k \cdot \omega_k$).

We study two different strategies in order to count the number of real roots :

- Hermite's method, which computes a quadratic form whose signature gives the number of real roots of the system.

– The Generalized Shape Lemma , which computes an univariate polynomial that has the same number of real roots as the system . The number of distinct real roots can then be computed using a Sturm-Habicht sequence.

Since the reduction of a quadratic form needs $O(n^3)$ basic arithmetic operations and Sturm-Habicht sequences needs $O(n^2)$ operations, the second method is asymptotically better. In practice however, the first method behaves better in many cases.

2. Hermite's method

For every $h \in A$, we define :

– The linear homomorphism of multiplication by h:

$$\begin{array}{rccc} m_h : & A & \rightarrow & A \\ & q & \longmapsto & h \cdot q \end{array}$$

– The matrix M_h of m_h with respect to $\mathcal{B}$
– The h-trace symmetric bilinear form (or simply trace if $h = 1$) :

$$\begin{array}{rccc} Tr_h : & A \times A & \rightarrow & K \\ & (f, g) & \longmapsto & Trace(fgh) \end{array}$$

where $Trace(fgh)$ is the trace of M_{fgh}.
– Hermite's quadratic form :

$$\begin{array}{rccc} Q_h : & A & \rightarrow & K \\ & f & \longmapsto & Tr_h(f, f) \end{array}$$

Let C (resp. R) be the algebraic closure (resp. real closure) of Z, the following theorem relates the rank and signature of Q_h to the number of zeros of S in C^n or R^n (see (Petersen & *al.*, 1993)) :

Theorem 1 *Let S be a zero-dimensional system of $Z[X_1, \ldots, X_n]$ with a finite set of distinct zeros $\mathcal{X}$, C (resp. R) the algebraic closure (resp. real closure) of Z. Then :*

– $rank(Q_h) = \sharp\{\delta \in C^n \bigcap \mathcal{X} \ , \ h(\delta) \neq 0\}$
– $signature(Q_h) = \sharp\{\delta \in R^n \bigcap \mathcal{X}, h(\delta) > 0\} - \sharp\{\delta \in R^n \bigcap \mathcal{X}, h(\delta) < 0\}$

Remark 1 *The second formula allows us to produce an algorithm "à la Ben Or - Kozen -Reif" to deal with polynomial inequalities ($= 0$, > 0 , < 0) over $X \bigcap R$ (see (*Ben-Or & *al.*, 1986*)).*

Applying the previous theorem with $h = 1$, we obtain an algorithm to compute the number of distinct real or complex roots of a zero-dimensional system :

Corollary 1 *With the notations of theorem 1,*

- $rank(Q_1) = \sharp\{\delta \in C^n \bigcap \mathcal{X}\}$
- $signature(Q_1) = \sharp\{\delta \in R^n \bigcap \mathcal{X}\}$

2.1. COMPUTING HERMITE'S QUADRATIC FORM

Let the multiplication table MT of A with respect to a linear basis $\mathcal{B} = \{\omega_1, \ldots, \omega_D\}$ be given and for $P \in A$, let $Vect(P)$ be the column-vector whose coordinates are the coefficients of P with respect to $\mathcal{B}$.

With these notations, the matrix $Q_{h,\mathcal{B}} = (Q_{h,\mathcal{B}}[i,j])_{1 \leq i,j \leq D}$ of Q_h with respect to the basis $\mathcal{B}$ is defined by :

$$Q_{h,\mathcal{B}}[i,j] = Q_h(\omega_i, \omega_j) = Trace(\omega_i \omega_j h) = \sum_{k=1}^{D} Vect(\omega_i \omega_j \omega_k h)_k$$

A naive algorithm would require us to compute all the products $\omega_i \omega_j \omega_k$ (resp. $\omega_i \omega_j \omega_k \omega_l$) to get $Q_{1,\mathcal{B}}$ (resp. $Q_{h,\mathcal{B}}$). Even if we can do these computations using the strategy proposed in (Rouillier , 1995), the method would be inefficient because of a dramatic growth of the number of monomials involved . In order to improve this, we use the linearity of the mapping trace, as much as possible :

Lemma 1 *If we denote by $Vtr(h)$ the column-vector*

$$[Trace(h\omega_1), \ldots, Trace(h\omega_D)]^T$$

then :

- $Vtr(h) = (Q_{1,\mathcal{B}})^T \cdot Vect(h)$
- *for all* $1 \leq i,j \leq D$, $Q_{h,\mathcal{B}}[i,j] = MT[i,j] \cdot Vtr(h)$

Proof : Let $h = \sum_{k=1}^{D} a_k \omega_k$. Then, for all $1 \leq i \leq D$,

$$Trace(h\omega_i) = \sum_{k=1}^{D} a_k Trace(\omega_k \omega_i)$$

and so $Vtr(h) = (Q_{1,\mathcal{B}})^T \cdot Vect(h)$.

Let $\omega_i \omega_j = \sum_{l=1}^{D} a_l^{(i,j)} \omega_l$, then :

$$\begin{aligned} Q_{h,\mathcal{B}}[i,j] &= \sum_{k=1}^{D} Vect(\omega_i \omega_j \omega_k h)_k = \sum_{k=1}^{D} \sum_{l=1}^{D} a_l^{(i,j)} Vect(\omega_l \omega_k h)_k \\ &= \sum_{l=1}^{D} a_l^{(i,j)} \sum_{k=1}^{D} Vect(\omega_l \omega_k h)_k = \sum_{l=1}^{D} a_l^{(i,j)} Trace(h\omega_l) \\ &= Vect(\omega_i \omega_j) \cdot Vtr(h) \end{aligned}$$

□

When studying systems with coefficients in Z, one obtains for Hermite's quadratic form a matrix with coefficients in K, but using a simple transformation, one can assume that the matrix of $Q_{h,\mathcal{B}}$ has coefficients in Z.

The author's recent algorithm (see (Rouillier, 1994)) generalizes the Bareiss identities in order to reduce $Q_{h,\mathcal{B}}$ by doing all the computations in Z with $O(D^3)$ basic operations, and allows a good control of the size of the intermediate results, when Z is the domain of integers (e.g $O(D(t+log(D)))$ if t is the maximum size of the coefficients of $Q_{h,\mathcal{B}}$).

We can now describe a first algorithm for computing the number of distinct real roots of a given zero-dimensional system :

Algorithm I

Input : A Groebner basis of S for any admissible monomial ordering and the associated multiplication table MT.

- **step 1 :** Compute $Vtr(1) = [\sum_{i=1}^{D}(MT[1,i])_i, \ldots, \sum_{i=1}^{D}(MT[D,i])_i]^T$
- **step 2 :** Compute

$$Q_{1,\mathcal{B}}[j,i] = Q_{1,\mathcal{B}}[i,j] = MT[i,j] \cdot Vtr(1) \ , \ 1 \leq i \leq j \leq D$$

- **step 3 :** Compute the signature of $Q_{1,\mathcal{B}}$.

3. Generalized Shape lemma

The idea of the Generalized Shape lemma (see (Alonso & *al.*, 1994)) is to express the solutions of a polynomial system as rational functions in the roots of a univariate polynomial.

Let $\mathcal{Y} = \{\beta_1, \ldots, \beta_D\}$ be the set of the roots (not necessary distinct) of S.

Let t be a separating element of A i.e. such that for every $x \neq y$ in $\mathcal{Y}$, $t(x) \neq t(y)$ and let v be any element of A.

Consider the polynomials

$$f(t,T) = \prod_{y \in \mathcal{Y}} (T - t(y))$$

$$g(v,t,T) = \sum_{x \in \mathcal{Y}} v(x) \prod_{y \in \mathcal{Y}, y \neq x} (T - t(y))$$

If β is a zero of S of multiplicity m, then $f(t,t(\beta)) = 0$, and we have

$$v(\beta) = \frac{g^{(m-1)}(v,t,t(\beta))}{g^{(m-1)}(t,t(\beta))} = \left(\frac{\frac{\partial^{m-1} g(v,t,T)}{\partial T^{m-1}}}{\frac{\partial^{m-1} g(t,T)}{\partial T^{m-1}}} \right)_{T=t(\beta)}$$

where $g(t,T) = f'(t,T) = g(1,t,T)$.

Proposition 1 *Let $S = \{f_1, \ldots, f_s\}$ be a zero-dimensional system of polynomials of $Z[X_1, \ldots, X_n]$, and $\mathcal{X} = \{\alpha_1, \ldots \alpha_d\}$ the distinct solutions of S with respective multiplicities $\{m_1, \ldots, m_d\}$.*

There exist a separating element t in A and polynomials

$$f(t,T)\ ,\ g(t,T)\ ,\ g_1(t,T),\ldots,g_n(t,T)$$

so that :

- *The roots of $f(t,T)$ are exactly $\{t(\alpha_1),\ldots,t(\alpha_d)\}$ with respective multiplicities $\{m_1,\ldots,m_d\}$*
- *If β is a zero of S of multiplicity m,*

$$X_i(\beta) = \frac{g_i^{(m-1)}(v,t,t(\beta))}{g^{(m-1)}(t,t(\beta))}$$

Proof : The existence of a separating element is given by the following lemma :

Lemma 2 *If $\mathcal{X}$ contains less than d points, then at least one among the $u_i = X_1 + iX_2 + \ldots + i^{n-1}X_n$ for $0 \leq i \leq (n-1)\frac{d(d-1)}{2}$ is a separating element.*

Proof : Consider a couple $(x,y) = ((x_1,\ldots,x_n),(y_1,\ldots,y_n))$ of distinct points of $\mathcal{X}$, and let $l(x,y)$ be the number of index i such that $u_i(x) = u_i(y)$. Since the polynomial $(x_1-y_1)+\ldots+(x_n-y_n)t^{n-1}$ has no more than $k-1$ distinct roots (it is not indentically null because $x \neq y$), $l(x,y)$ is less than $k-1$. Since the total number of couples of distinct points of $\mathcal{X}$ is less than $\frac{d(d-1)}{2}$, this complete the proof. □

Given a separating element t, we complete the proof of the proposition by taking $g_i(t,T) = g(X_i,t,T)\ ,\ i = 1\ldots n$. □

According to the previous proposition, the number of distinct real roots of a given system can be easily computed from the Generalized Shape Lemma :

Corollary 2 *Let t be a separating element of A. The number of distinct real roots of S is exactly the number of real roots of $f(t,T)$. It can also be found by computing the Sturm-Habicht sequence of $f(t,T)$*

We will not discuss here the problem of finding a separating element, but, given $f(t,T)\ ,\ g(t,T)\ ,\ g_1(t,T),\ldots,g_n(t,T)$ for any t (randomly chosen) , we assume that there exists a simple method to check if t is a separating element or not (see (Rouillier , 1995)).

An algorithm that computes the number of distinct real roots of a given zero dimensional system can now be described :

Algorithm II

Input : A Groebner basis of S for any admissible monomial ordering and the associated multiplication table MT.

- **step 1 :** Take any t among

$$\{X_1 + iX_2 + \ldots + i^{n-1}X_n \ , \ i = 0 \ldots (n-1)\frac{D(D-1)}{2}\}$$

- **step 2 :** Compute $f(t,T)$, $g(t,T)$, $g_1(t,T), \ldots, g_n(t,T)$
- **step 3 :** Check if t is a separating element, and if not go to **step 1**.
- **step 4 :** Compute the Sturm-Habicht sequence of $f(t,T)$.

3.1. COMPUTING GSL USING TRACES AND SYMMETRIC FUNCTIONS

The method we propose for computing $f(t,T) = \prod_{y \in \mathcal{Y}}(T - t(y))$ uses the classical notion of elementary symmetric functions and their connection with Newton sums.

Notation 1 *We denote by :*

- $S_i(t,\mathcal{Y}) = \sum_{\mathcal{I} \subset \mathcal{Y} \ \sharp\mathcal{I}=i} \prod_{y \in \mathcal{I}} t(y)$ *the* i^{th} *elementary symmetric function associated with* $\{t(\beta_1), \ldots, t(\beta_D)\}$.
- $N_i(t,\mathcal{Y}) = \sum_{y \in \mathcal{Y}} t(y)^i$ *the* i^{th} *elementary Newton sum associated with* $\{t(\beta_1), \ldots, t(\beta_D)\}$.

According to this notation, the classical relations that link elementary symmetric functions to elementary Newton sums can be written as follows :

$$(D-i)S_i(t,\mathcal{Y}) = \sum_{j=0}^{i}(-1)^j N_j(t,\mathcal{Y})S_{i-j}(t,\mathcal{Y})$$

with the convention $S_0(t,\mathcal{Y}) = 1$.

Let M_t the matrix of multiplication by a polynomial t in A. Since the eigenvalues of M_t are the scalars : $\{t(y) \ , \ y \in \mathcal{Y}\}$ (see (Petersen & *al.*, 1993)), the set $\{N_j(t,\mathcal{Y})\}$ can be computed using the relation :

$$Trace(t^i) = N_i(t,\mathcal{Y})$$

Since $S_i(t,\mathcal{Y})$ is the coefficient of T^{D-i} in the polynomial $\prod_{y \in \mathcal{Y}}(T - t(y))$, $f(t,T)$ can be easily computed by using the traces $Trace(t^i)$, $i = 1, \ldots D$.

Expanding the polynomial $g(v,t,T)$ (of degree $D-1$), we note that the coefficient of T^{D-i-1} in $g(v,t,T)$ is

$$(-1)^i \sum_{y \in \mathcal{Y}} v(y)S_i(t,\mathcal{Y} \setminus \{y\})$$

We also extend the notion of elementary symmetric function and elementary Newton sums :

Definition 1 *Given two polynomials t and $v \in K[X_1, \ldots, X_n]$ and $\mathcal{Y} \in C^n$,*

- $S_i(v, t, \mathcal{Y}) = \sum_{y \in \mathcal{Y}} v(y) S_i(t, \mathcal{Y} \setminus \{y\})$
- $N_i(v, t, \mathcal{Y}) = \sum_{y \in \mathcal{Y}} v(y) t(y)^i$

Lemma 3 *Generalized elementary Newton sums and classical elementary symmetric functions can be linked using the following formula : For $0 \leq k < i$,*

$$\begin{array}{rl} \sum_{y \in \mathcal{Y}} v(y) t(y)^k S_{i-k}(t, \mathcal{Y} \setminus \{y\}) = & N_k(v, t, \mathcal{Y}) S_{i-k}(t, \mathcal{Y}) - \\ & \sum_{y \in \mathcal{Y}} v(y) t(y)^{k+1} S_{i-k-1}(t, \mathcal{Y} \setminus \{y\}) \end{array}$$

Proof : For all k, $0 \leq k < i$,

$$\begin{array}{rcl} N_k(v,t,\mathcal{Y}) S_{i-k}(t,\mathcal{Y}) & = & \left(\sum_{y \in \mathcal{Y}} v(y) t(y)^k\right) \left(\sum_{I \subset \mathcal{Y}\, ,\, \sharp I = i-k} \prod_{z \in I} t(z)\right) \\ & = & \sum_{y \in \mathcal{Y}} \sum_{I \subset \mathcal{Y}\, ,\, \sharp I = i-k} v(y) t(y)^k \prod_{z \in I} t(z) \\ & = & \sum_{y \in \mathcal{Y}} (\sum_{I \subset \mathcal{Y}\, ,\, \sharp I = i-k} v(y) t(y)^k \prod_{z \in I\, ,\, z \neq y} t(z) \\ & & + \sum_{I \subset \mathcal{Y}\, ,\, \sharp I = i-k-1} v(y) t(y)^{k+1} \prod_{z \in I\, ,\, z \neq y} t(z)) \\ & = & \sum_{y \in \mathcal{Y}} v(y) t(y)^k S_{i-k}(t, \mathcal{Y} \setminus \{y\}) + \\ & & \sum_{y \in \mathcal{Y}} v(y) t(y)^{k+1} S_{i-k-1}(t, \mathcal{Y} \setminus \{y\}) \end{array}$$

$\square$

We now extend the relation between Newton sums and symmetric functions :

Proposition 2

$$S_i(v, t, \mathcal{Y}) = \sum_{j=0}^{i} (-1)^j N_j(v, t, \mathcal{Y}) S_{i-j}(t, \mathcal{Y})$$

Proof According to lemma 3, we have :

$$\begin{array}{rcl} S_i(v,t,\mathcal{Y}) & = & \sum_{y \in \mathcal{Y}} v(y) S_i(t, \mathcal{Y} \setminus \{y\}) \\ & = & N_0(v,t,\mathcal{Y}) S_i(t,\mathcal{Y}) - \sum_{y \in \mathcal{Y}} v(y) t(y) S_{i-1}(t, \mathcal{Y} \setminus \{y\}) \end{array} \quad B$$

Using the same argument, we obtain by induction :

$$\begin{array}{rcl} S_i(v,t,\mathcal{Y}) & = & \sum_{j=0}^{i-1} (-1)^j N_j(v,t,\mathcal{Y}) S_{i-j}(t,\mathcal{Y}) \\ & & + (-1)^i \sum_{y \in \mathcal{Y}} v(y) t(y)^i S_0(t, \mathcal{Y} \setminus \{y\}) \end{array}$$

Since $S_0(t, \mathcal{Y} \setminus \{y\}) = 1$ the proof is complete. $\square$

Since $Trace(vt^i) = N_i(v, t, \mathcal{Y})$ (for $y \in \mathcal{Y}$, the scalars $vt^i(y)$ are the eigenvalues of M_{vt^i}) $g(v, t, T)$ can be deduced from $f(t, T)$ and the traces $Trace(vt^i)$, $i = 1 \ldots (D-1)$.

According to the previous part, there is an easy way to compute the generalized Shape Lemma from the traces : $Trace(vt^i)$, $i = 1, \ldots, D-1$ and $Trace(t^i)$, $i = 1, \ldots, D$.

Assume that all the products $Vect(\omega_i \omega_j)$ are known. A straightforward algorithm computes all the needed products vt^i and, using the previous results, the expressions $Trace(vt^i)$.

Procedure Compute-Traces-I
Input :
MT
t /* a separating element */
Output :
$Newton_t$, a vector of dimension $D+1$ so that $Newton_t[i] = N_i(t, \mathcal{Y})$
Tr_t, a $n \cdot D$ matrix so that $Tr_t[i,j] = N_j(X_i, t, \mathcal{Y})$
Begin
$Vtr := [Trace(\omega_1), \ldots, Trace(\omega_D)]^T$
$tmp := [1, 0, \ldots, 0]$
For j from 0 to $D-1$ do
$Newton_t[j] := tmp \cdot Vtr$
For i from 1 to n do
$Tr_t[i,j] := (M_{X_i} \cdot tmp) \cdot Vtr$
If $j < D-1$
$tmp := M_t \cdot tmp$
$Newton_t[D] := tmp \cdot Vtr$

End

Proof of the algorithm Since $Vect(P) \cdot Vtr = Trace(P)$, the proof of the algorithm is obvious. □

The following procedure is an optimization of the previous one, significantly decreasing the number of basic operations :

Procedure Compute-Traces
/* Same Input and Output as Compute-Traces-I */
Begin
$tmp := [Trace(\omega_1), \ldots, Trace(\omega_D)]^T$
$Newton_t[0] := D$
For j from 0 to $D-1$ do
For i from 1 to n do
$Tr_t[i,j] := Vect(X_i) \cdot tmp$
$Newton_t[j+1] := Vect(t) \cdot tmp$
$tmp := (M_t)^T \cdot tmp$
End

Proof of the algorithm If we notice that

$$Vect(P) \cdot [Trace(Q\omega_1), \ldots, Trace(Q\omega_D)] = Trace(PQ)$$

then, after the j^{th} step in the principal loop :

$$tmp = [Trace(u^j\omega_1), \ldots, Trace(u^j\omega_D)]^T$$

and so, $\begin{cases} Newton_t[i] = N_i(t, \mathcal{Y}) \ , \ i = 1, \ldots, n \\ Tr_t[i,j] = N_j(X_i, t, \mathcal{Y}) \end{cases}$

□

4. Benchmarks

In this section, we use examples from robotics (e.g. direct kinematic problem of a parallel robot, see (Ditrit & *al.*, 1995), (Faugère and Lazard, 1994), (Merlet, 1993), ...) in order to compare the two methods described before.

Legend :

- GSL : computation of the Generalized Shape Lemma.
- St-Habicht : Sturm-Habicht's algorithm (applied on the univariate polynomial given by GSL).
- Hermite : Hermite's algorithm.
- Roots :
  - R : number of distinct real roots
  - C : number of distinct complex roots
- T. : computation time in seconds on a Sun-Sparc 10Mhz/128Mo, using the PoSSo-Library.
- L. : binary length of the largest coefficient that appears in the result.

We assume that GSL and Hermite algorithms take as input a Groebner basis (computed with respect to the *Degre Reverse Lexicographic* monomial ordering) and the associated multiplication table.

	Alg. II				**Alg. I**		Roots	
	GSL		St-Habicht		Hermite			
Name	T.	L.	T.	L.	T.	L.	R	C
kin1 (Ditrit)	7	8	63	444	22	132	8	40
kin2 (Merlet)	247	87	591	1226	457	1647	16	36
planN3I (Faugère)	25	32	198	936	179	654	0	40
spatial1 (Faugère)	51	30	44	815	15	590	0	16
spatial2 (Innocenti)	1236	77	4803	3514	5015	2944	24	40
spat2A2I (Faugère)	212	38	626	1839	316	894	0	32
spat66 (Faugère)	1935	40	2006	2852	7955	2348	2	40

5. Conclusion

Even if we do not take care of the preprocessing that is needed in order to compute an univariate representation of the systems, Hermite's method is more efficient in most cases. Since Sturm - Habicht sequences need $O(n^2)$ basic operations and the quadratic form reduction needs $O(n^3)$ operations, this result is due to a better control of the size of the coefficients in the algorithm that reduces the quadratic forms.

References

Alonso, M.E., Becker, E., Roy, M.F., Wörmann, T.(1994) Zero's, Multiplicities and Idempotents for Zero Dimensional Systems, *MEGA 94* **to be published**.

Ben-Or, M., Kozen, D., and Reif, J. (1986) Complexity of Elementary Algebra and Geometry *Journal of Computation and Systems Sciences* **Vol. 32 p. 251-264**.

Ditrit, O., Petitot, M., Walter, E. (1995) Guaranteed numerical solution of sets of nonlinear equations by Hansen's algorithm, implementation and applications *Proceedings of the PoSSo Workshop on Software* **p. 19-34**.

Faugère, J.C. (1994) Résolution des systèmes d'équations polynômiales *Doctoral Thesis.*

Faugère, J.C., Gianni, P., Lazard, D. and Mora, T. (1994) Efficient Computation of Zero-dimensional Gröbner Basis by Change of Ordering. *J. Symbolic Computation*

Faugère, J.C. and Lazard, D. (1994) The combinatorial classes of parallel manipulators *Mechanism and Machine Theory.*

Merlet, J-P. (1990) Les Robots Parallèles *Hermès,Paris.*

Pedersen, P. Roy, M.F, Szpirglas, A. (1993) Counting Real Zeros in the Multivariate Case *Computational Algebraic Geometry, Frédéric Eyssette, André Galligo (editors), Birkhäuser* **p. 203-223**.

Rouillier, F. (1994) Formules de Bareiss et réduction de formes quadratiques *Notes aux Comptes Rendus de l'Académie des Sciences* **To be published**.

Rouillier, F. (1995) PoSSo-RealSolving (Zero-dimensional systems of polynomials) *Proceedings of the PoSSo Workshop on Software* **p. 125-151**.